AMENDMENTS TO THE SPECIFICATION:

Please replace the brief description of the drawings labeled paragraph 0005 with the following paragraph/list:

[0005] Figure 1A depicts an SCI system arranged in a ring topology interconnection.
Figure 1B depicts an SCI system arranged in a dual-ring topology with data flow in two directions.
Figure 1C depicts an SCI system interconnected by an SCI switch.
Figure 1D depicts a dual-axis, ring based SCI interconnection network.
Figure 2A depicts a two drive RAID 0 configuration.
Figure 2B depicts a two drive RAID 1 configuration.
Figure 2C depicts a seven drive RAID 2 configuration.
Figure 2D depicts a three drive RAID 4 configuration.
Figure 2E depicts a three drive RAID 5 configuration.
Figure 2F depicts a four drive RAID 6 configuration.
Figure 2G depicts a four drive RAID 1+0 configuration.
Figure 3A depicts a hardware RAID controller connected to a processor and two disk drives.
Figure 3B depicts a software RAID controller on a processor connected to two disk drives.
Figure 4A depicts a RAID system having redundancy in two planes.
Figure 4B depicts a RAID system having redundancy in two planes and facilities for HSM segment journaling and tape backup.
Figure 4C depicts a dual-planar parity RAID system being configured to provide two RAID 1 HSM systems.
Figure 4D depicts a dual-planar parity RAID system being configured to provide one RAID 1 HSM system with three virtual mirrored drives.

Figure 5 illustrates the advantages of a dual-planar parity RAID system with respect to reliability and recovery of data in the event of drive failure.

Figure 6 illustrates the advantages of a dual-planar parity RAID system configured to provide two RAID 1 virtual drives with respect to reliability against drive failure.

Figure 7 depicts a simple RAID network subsystem that may provide RAID storage to clients.

Figure 8 depicts the components of one implementation of a RAID network subsystem.

Figures 9A and 9B depict the front and rear views of one implementation of a RAID network subsystem expansion cabinet.

Figure 10 depicts one interconnection configuration of RNS expansion cabinets through an SCI switch to four clients using several protocols.

Figure 11 depicts one interconnection configuration of RNS expansion cabinets through an Ethernet network to four clients using several protocols.

Figure 12 depicts the logical connections of four clients to an RNS expansion cabinet through several protocols.

Figure 13 depicts a configuration providing a DASD string using RAID 5+1 with failover and breakaway mirror capabilities.

Figure 14 depicts several Linux client configurations of connections to RNS devices of several configurations.

Figure 15 depicts several Windows 2000 client configurations of connections to RNS devices of several configurations.

Figure 16 depicts the several parts of a hot-fixable disk partition or segment.

Figure 17 depicts the configuration of RAID segments within an M2CS partition.

Figure 18 depicts a logical combination of two M2CS partitions to form two virtual disks.

Figure 19 depicts the hotfix table area of a segment or partition.

Figure 20 depicts a hotfix recovery method using a RAID 1+0 system.

Figure 21 depicts a hotfix recovery method using a RAID 5 system.

Figure 22 depicts a method of distributing read operations to a mirrored RAID array.

Figure 23 depicts a method of re-mirroring a RAID 1 array for inclusion.

Figure 24 depicts snapshots of a journalled disk.

Figure 25 depicts a method of recording disk snapshots to a journal of journal segments.

Figure 26 depicts real time infinite segment journaling on a dual-disk system.

Figure 27 depicts disaster recovery using an infinite HSM segmented journal.

Figure 28 depicts the use of a checkpoint bitmap to determine the finality of a disaster recovery using an infinite HSM segmented journal.

Figure 29A illustrates a real time journaling method.

Figure 29B illustrates a snapshot journaling method.

Figure 30 depicts the operation of a real time infinite segment journaling system having two virtual RAID disks.

Figure 31 depicts a tape master catalog with respect to tape segments.

Figure 32 depicts an RNS system that includes the Linux operating system.

Figure 33 depicts an LRU cache, a dirty LRU list, a disk elevator, and process queues forming part of an RNS system.

Figure 34 depicts the operation of a compressed LRU cache.

Figure 35 illustrates basic physical components of a disk drive.

Figure 36 depicts the operation of an extensible hash in a disk elevator.

Figure 37 illustrates some of supported block sizes for Linux for various filesystem types.

Figure 38 illustrates some of supported cluster sizes for Windows NT and 2000 for various filesystem types.

Figure 39 depicts a method of performing read-ahead on a RAID 1 system.

Figure 40A depicts a read-ahead method using read requests as input.

Figure 40B depicts a read-ahead method using filesystem knowledge and read requests as input.

Figure 41 depicts one method of dynamic block stripe allocation.

Figure 42 depicts an RNS client system that includes the Linux operating system.

Figure 43 depicts an RNS client system that includes the Windows NT or Windows 2000 operating systems.

Figure 44 depicts another RNS system that includes the Linux operating system.

Figure 45 depicts a system combining four RNS systems through an SCI switch to a client.
Figure 46A depicts a simple NORMA parallel computing system.
Figure 46B depicts a simple NUMA parallel computing system.
Figure 46C depicts a simple CCNUMA parallel computing system.

Immediately following paragraph 0002, please replace the heading "BACKGROUND OF THE INVENTIONS" with "BACKGROUND OF THE INVENTION".

Immediately following paragraph 0003, please replace the heading "BRIEF SUMMARY OF THE INVENTIONS" with "BRIEF SUMMARY OF THE INVENTION".

Immediately following paragraph 0006, please replace the heading "DETAILED DESCRIPTION" with "DEFINITIONS".

Immediately following paragraph 0022 and before paragraph 0023, please insert the heading "DETAILED DESCRIPTION".

Please place the following paragraphs in the place of the existing paragraphs, each identified by paragraph number:

[0001] High Speed Fault Tolerant Storage System

[0003] In 1992, the Institute for Electronics and Electrical Engineers (IEEE) established a standard for a scalable coherent interface {SCI}. SCI supports distributed multiprocessing at high bandwidth, with low latency, and providing scalable architecture. Since the 1980's, there has been continuous development of redundant arrays of inexpensive or independent drives (RAID) which provide fault-tolerant data storage and access capabilities. It is against the general background of RAID and SCI that the invention is made.

[0004] The invention relates generally to high-speed fault tolerant storage systems. More particularly, the invention relates to fault tolerant storage systems utilizing RAID and/or SCI, combinations thereof, and features usable with one of both of them, or otherwise usable in a data storage or data access environment. Detailed information on various example embodiments of the invention are provided in the Detailed Description below, and the invention is defined by the appended claims.

[0006] Reference will now be made in detail to some embodiments of the invention, example of which are illustrated in the accompanying drawings.

[0009] In the industry there have become defined several levels of RAID systems. The first level, RAID-0, combines two or more drives to create a larger virtual disk. In the dual drive RAID-0 system is illustrated in figure 2A one disk 200 contains the low numbered sectors or blocks and the other disk 202 contains the high numbered sectors or blocks, forming one complete storage space. RAID-0 systems generally interleave the sectors of the virtual disk across the component drives, thereby improving the bandwidth of the combined virtual disk. Interleaving the data in that fashion is referred to as striping. RAID-0 systems provide no redundancy of data, so if a drive fails or data becomes corrupted, no recovery is possible short of backups made prior to the failure.

[0010] RAID-1 systems include one or more disks that provide redundancy of the virtual disk. One disk is required to contain the data of the virtual disk, as if it were the only disk of the array. One or more additional disks contain the same data as the first disk, providing a 'mirror' of the data of the virtual disk. A RAID-1 system will contain at least two disks, the virtual disk being the size of the smallest of the component disks. A disadvantage of RAID-1 systems is that a write operation must be performed for each mirror disk, reducing the bandwidth of the overall array. In the dual drive RAID-1 system of figure 2B, disk 204 and disk 206 contain the same sectors or blocks, each disk holding exactly the same data.

[0011] RAID-2 systems provide for error correction through hamming codes. The component drives each contain a particular bit of a word, or an error correction bit of that word. Figure 2C, for example, illustrates a RAID-2 system having a constituent word of 4 bits and 3 hamming code error correction bits. Disks 208, 210, 212, and 214 contain bits 0, 1, 2, and 3 of each word of storage, while disks 216, 218 and 220 contain the hamming error correction bits. RAID-2 systems automatically and transparently detect and correct single-bit defects, or single drive failures, while the array is running. Although RAID-2 systems improve the reliability of the array over other RAID types, they are less popular than some other systems due to the expense of the additional drives, and redundant onboard hardware error correction.

[0012] RAID-4 systems are similar to RAID-0 systems, in that data is striped over multiple drives, as exemplified by the three disk RAID-4 system of figure 2D. The storage spaces of disks 222 and 224 are added together in interleaved fashion, while disk 226 contains the parity of disks 222 and 224. RAID-4 systems are unique in that they include an additional disk containing parity. For each byte of data at the same position on the striped drives, parity is computed over the bytes of all the drives and stored to the parity disk. The XOR operation is used to compute parity, providing a fast and symmetric operation that can regenerate the data of a single drive, given that the data of the remaining drives remains intact. RAID-3 systems are essentially RAID-4 systems with the data striped at byte boundaries, and for that reason RAID-3 systems are generally slower than RAID-4 systems in most applications. RAID-4 and RAID-3 systems therefore are useful to provide virtual disks with redundancy, and additionally to provide large virtual drives, both with only one additional disk drive for the parity information. They have the disadvantage that the data throughput is limited by the throughput of the drive containing the parity information, which must be accessed for every read and write operation to the array.

[0013] RAID-5 systems are similar to RAID-4 systems, with the difference that the parity information is striped over all the disks with the data, as exemplified by the three disk system of figure 2E. Disks 228, 230, and 232 each contain data and parity in interleaved fashion. Distributing the parity data generally increases the throughput of the array as compared to a RAID-4 system. RAID-5 systems may continue to operate though one of the disks has failed. RAID-6 systems are like RAID-5 systems, except that dual parity is kept to provide for normal operation if up to the failure of two drives. An example of a RAID-6 system is shown in figure 2F. Disks 234, 236, 238, and 240 each contain data and two parity words labeled P and R.

[0014] Combinations of RAID systems are also possible. For example, figure 2G illustrates a four disk RAID 1+0 system providing a concatenated file system that is also redundant. Disks 242 and 244 are mirrored, as are 246 and 248. The combination of 242 and 244 are added to 246 and 248 to form a storage space that is twice the size of one individual drive, assuming that all four are of equal size. Many other combinations of RAID systems are possible.

[0015] Many implementations of RAID controllers exist, two types being typical. The first common RAID controller, exemplified in figure 3A, is hardware based, having a disk controller interfaced to a computer and several other disk controller interfaced to two or more disk drives. A processor, or other device requiring access to a RAID virtual disk is connected to an adapter 302 through an interface. Interface 304 may be a bus interface, such as PCI, or it may be a disk interface, thereby connecting the adapter as a standard disk drive. Common interfaces of today's computers are the IDE interface and the SCSI interface. Adapter 302 interfaces to disks of the RAID array, shown as 310 and 312, through other disk interfaces, shown as 306 and 308. Adapter 302 contains logic or other RAID means to present the array of disks to the processor 300 as a single disk. Hardware-based RAID controllers have the advantage of speed. The second common RAID controller is software based, exemplified in figure 3B. In that case, the controller is a part of the operating system of the computer, the computer having interfaces to two or more drives. A processor 314 contains software to perform RAID functions and also has disk interfaces, shown as 316 and 318, to which two or more disks, shown as 320 and 322, are accessible to the RAID software. Software based controllers are more economical than hardware based controllers, but consume a portion of the processor capacity to perform the RAID functions.

[0016] The following definitions are common to the field of parallel computing and will be pertinent to some implementations of the invention herein. A NORMA (NO Remote Memory Access) system, illustrated in figure 46A, lacks direct hardware facilities for the access of one processor's memory by another processor in an interconnection network. Most computers fall into this category. Parallel computing may be performed over a network 2600 by message passing, provided that the processor 4604 is connected by a network interface 4604 to the network 4600. Those systems maintain all memory locally 4602. NORMA systems may provide access of memory by software applications and drivers. NORMA systems are generally unsuited for parallel computing applications due to the low bandwidths and high latencies of data transfers between processors. NUMA (Non-Uniform Memory Access) systems, illustrated in figure 46B, provide facilities for memory access of one processor to another by hardware. NUMA systems, in their fashion, present to software shared memory such that an application may access memory attached to a distant processor as if it were local to the processor on which the application is executing. The characteristics of a NUMA system are generally a processor 4624 having local memory 4622, but also having access to shared memory 4620 by way of a shared memory controller. The shared memory 4620 is virtual, in that it appears to be a distinct entity to processors 4624, but is actually maintained at processors in the parallel network. NUMA systems in general do not provide a memory cache of the shared memory, thus each memory access requires the full data of each read or write be passed across the interconnection fabric. CCNUMA (Cache Coherent NUMA) systems, illustrated in figure 46C, do provide a hardware cache of shared memory, thus eliminating the need of passing blocks of memory data across the interconnection network when the cache is coherent with the shared memory on the remote processor (and thus is coherent). The characteristics of a CCNUMA system are a processor 4644 having local memory 4642, also having access to shared memory through a shared memory controller 4646, but with caching facilities having the capability of tracking and maintaining the coherency of the cache with the other controllers on the shared memory network.

[0017] The Scalable Coherent Interface (SCI) is an interconnection scheme described by the standard IEEE 1596 (1992). SCI was originally intended for parallel computing applications to provide a high-speed communications channels between processors. SCI links are unidirectional to eliminate the need for bus arbitration and inherent latencies. An SCI interconnected system is generally arranged in a ring topology, as shown in figure 1A. In such a system data is passed from one processor element or node to the next until the data reaches its intended recipient. A multi-processor system arranged in a ring topology is susceptible to failure should any single failure occur in any communications link or node. To reduce the risk of failure, a multi-processor system may be implemented with a topology of multiple rings, as shown in figure 1B. A multi-processor system may also contain a switch, as shown in figure 1C, which may permit a part of the multi-processor system to continue operation should a node or a link fail. An SCI topology may also contain two or more axes, forming an interconnection fabric, as shown in figure 1D. Providing more than one topological axis also creates redundancy, while making routing of information somewhat more complex.

[0023] One embodiment of the invention includes a RAID subsystem having two planes, as exemplified by the system of figure 4A. Three or more disks form RAID-5 arrays in the X-plane, one example including 5 disks shown as 401, 402, 403, 404, and 405. Each of the disks of the X-plane RAID arrays are located in a unique Y-plane location, an example of a Y-plane being 402, 412, 422, 432, and 442. Each X-plane array is included as a virtual disk in the Y-plane, forming a RAID array at levels 0, 1, 1+0, 5, or other configuration as will be understood by those skilled in the art. Y-plane parity disks 441, 442, 443, and 444 provide for redundancy of data of the disks located in the same Y-axis, thereby forming a dual X-Y fault tolerant fabric preferably composed of inexpensive IDE and/OR SCSI disk devices. In a preferred embodiment, disks 405, 414, 423, and 432 contain the X-plane checksum data. Figure 4B illustrates a related embodiment to that of figure 4A, wherein HSM Disk cache storage segment journals are provided as 406, 407, 416, 417, 426, 427, 436, and 437 to tape drives 408, 418, 428, and 438.

[0024] A related embodiment of the system of figure 4B is shown in figure 4C. Drives 450, 451, 452, 453, and 454 are combined in a RAID array, as are drives 460, 461, 462, 463, and 464. Drives 455, 456, 457 458, and 459 are configured as mirrors in RAID-1 fashion to drives 450-454, as are drives 465, 466, 467, 468, and 469 to drives 460-464. Parity drives 478, 479, 480, 481, and 482 provide a Y-axis checksum for the entire array, for example 478 contains parity for drives 450, 455, 460, and 465. HSM disk cache storage segment journals 470 and 471 and tape drives 474 and 475 are provided for the cluster of drives 450-454 to provide HSM functions. Likewise HSM cache storage segment journals 472 and 473 and tape drives 476 and 477 are provided for the cluster of drives 460-464. A parity disk, such as 482, may be omitted from the system, if a connection is not available, as the drive adds only minimal amounts of redundancy to the system.

[0025] Another related embodiment of the system of figure 4B is shown in figure 4D. Drives 490a, 490b, 490c, 490d and 490e are combined in a RAID array. Drives 491a, 491b, 491c, 491d and 491e are configured as mirrors in RAID 1 fashion to drives 490a-e, as are drives 492a, 492b, 492c, 492d and 492e, and 493a, 493b, 493c, 493d and 493e to the same drives 490a-e. Parity drives 496a, 496b, 496c, 496d and 496e provide a Y-axis checksum for the entire array, for example 496a contains parity for drives 490a, 491 a, 492a and 493a. HSM disk cache storage segment journals 494a and 494b are provided for the cluster of drives 490a-490e to provide HSM functions. A parity disk, such as 496e, may be omitted from the system, if a connection is not available, as the drive adds only minimal amounts of redundancy to the system.

[0031] Use of RAID 4 Y-parity planes allows RAID 1+0 configurations to recover from failures of multiple mirrored devices, increasing the fault tolerance over traditional RAID 1+0 array configurations. The example of figure 6 shows failures in an array of RAID 1+0 devices, as was shown in figure 4C. Drives 600-604 and 620-624 form two raid arrays, with mirror drives 610-614 duplicating the data of drives 600-604 and mirror drives 630-634 duplicating the data of drives 620-624. Y-axis parity disks 640-644 contain parity for the drives in the same Y-axis, for example drive 640 containing parity for drives 600, 610, 620, and 630. In the example of figure 6, failures of both the primary and secondary mirrored devices in a RAID 1+0 array can be recovered from the Y plane RAID 4 parity check disks in real time. The failure of disks 600 and 610, for example, can be recovered because 600 and 610 contain duplicate data and 620/630 and parity disk 640 contain the remaining data for reconstruction.

[0035] In that embodiment, an RNS is housed in an expansion cabinet mountable to a 19 inch rack by slide rails that permits up to eight of those RNS devices to be inserted into a chassis forming a single DASD assembly. Drive bays are accessible by sliding the unit out of the cabinet and adding or removing drives into empty drive bays in the unit. The cabinet assembly allows hot swappable drives to be added to the RNS expansion cabinet. In that embodiment, shown in fig. 8, each RNS expansion cabinet is equipped with two redundant power supplies 802a and 802b having two cooling fans each, two PCI-SCI interface adapters 804a and 804b providing dual ring or single ring SCI topologies, a Pentium II or Pentium III mezzanine card 806 with a PCI bus 808, two SCSI II on-board controllers 810a and 810b, four 3Ware single-ended 8-way IDE controllers or four SCSI II controllers 812a, 812b, 812c and 812d, either 256 or 512MB of RAM, 32 80GB IDE or SCSI disk drives 814 (each individually labeled 815), a 10-100-1000 Ethernet interface 816, and a 19 inch cabinet chassis 800, two serial ports 818a and 818b, either 4MB or 8MB of flash memory 820 to hold the operating system, and four 32GB tape drives. The chassis 800 has two sliding rails, an array of 64 and an array of 8 LED displays for showing disk power or activity mounted to the front panel. HSM segment journal cache disks are optional. Each cabinet supports dual redundant power supplies with two cooling fans each, in the event of a single power supply failure. The mezzanine PII/PIII system board may contain integrated video, serial, and SCSI and IDE support on-board.

[0037] RNS expansion cabinets can be assembled into strings of SCI based DASD RAID and mirrored storage. In one embodiment, a single DASD cabinet of RNS storage can accommodate eight RNS expansion cabinets, each with 32 80GB disk drives providing 2.56 terabytes, for a total of 20.48 terabytes of available disk storage per vertical DASD cabinet. Figure 9A shows a front view of such a cabinet 900, and figure 9B shows a rear view of that same embodiment. In that embodiment each expansion cabinet 901 has drive indicator LEDs 902 indicating power and disk activity for a particular disk device in the array. Within a DASD cabinet RNS expansion cabinets are interconnected via SCI cables through two SCI ports, 904a and 904b. Optionally, each RNS expansion cabinet can also be connected to a standard Ethernet network through Ethernet port 906 and accessed remotely via TCP/IP, IPX/SPX, or UDP (NFS). In that embodiment, each of the eight RNS expansion cabinets have a parallel port 908, two serial ports 910a and 910b, two cooling fans 912a and 912b, and power receptacles 914a and 914b for dual power supplies.

[0038] RNS DASD cabinets can be interconnected via SCI into very large networks of thousands of RNS nodes, as shown in figure 10. DASD strings of RNS expansion cabinets 1010, 1012, and 1014 are linked by an SCI network pivoting around SCI switch 1008. SCI devices 1000, 1002, 1004, and 1006 access DASD strings 1010, 1012 and 1014 through the SCI network. Each of DASD strings 1010, 1012 and 1014 is composed of DASD cabinets labeled 1001. An Ethernet connection 1016 is provided by string 1010 in order to provide access to external devices to the DASD string through an Ethernet network. An RNS expansion cabinet can also be configured as a "head of string" and combine the adjacent RNS cabinets into large logical arrays of mirrored storage that are segregated into distinct SCI rings and ringlets. This provides the advantage of allowing host systems the ability to access several strings of SCI storage in tandem. If one of the strings fails, the SCI attached host can fail over to a mirrored string of DASD. DASD string mirroring is another level of mirroring provided by this invention and may be implemented at the SCI host's system adapter software allowing an SCI attached host system to select and mirror to one or more DASD strings in a distributed fashion.

[0039] RNS DADS cabinets or DASD strings can also be used as large Network Attached Storage servers to provide Ethernet based storage area networks. An RNS can also be configured to host multiple NFS, NCP, SMBFS, or iSCSI clients via an Ethernet connection, as shown in figures 11 and 12. In figure 11, DASD strings 1108, 1110, and 1112 are connected to an Ethernet network. As in figure 10, DASD strings 1108, 1110 and 1112 is composed of DASD cabinets labeled 1101. A client 1100 accesses the DASD strings through the NCP protocol. Another client 1102 accesses the DASD string through the NFS protocol. A third client 1104 accesses the DASD strings through the iSCSI protocol. A fourth client 1105 accesses the DASD strings through the NCP, NFS, and iSCSI protocols. In figure 12, a DASD string 1208 (again, composed of DASD cabinets labeled 1201) is made available through Ethernet connections 1200, 1202, 1204, and 1206 to a client. The first connection 1200 accesses the string 1208 using the iSCSI protocol, the storage being accessible under /dev/sda and /dev/sdb. The second connection 1202 accesses the string 1208 using the NCP protocol, the storage being accessible under SYS:/ and VOL1:/. The third connection 1204 accesses the string 1208 using the NFS protocol, the storage being accessible under /mnt/remote. The fourth connection 1206 also accesses the string 1208 using the NFS protocol, the storage being accessible under /mnt/remote1.

[0108] Figures 29A and 29B illustrate the difference between real time journalling and snapshot journalling in an infinite HSM journal. Three changes are made to the same block on a virtual disk 2900, 2901 and 2902. In figure 29A real time journalling is used. As each change occurs a new record is written to journal segment 2910, thereby illustrating that all incremental changes to the virtual disk are written to the journal segments. In figure 29B snapshot journalling is used, with a time domain boundary 2903 occurring between snapshots 2901 and 2902. The first journal segment 2920 recorded events before the time domain boundary 2903, but new records overwrite older records of the same block. Thus the earlier change of 2900 does not appear in 2920, but the last change of 2901 does. After the segment journal 2920 is closed at time domain boundary 2903, new change records are written to the successive segment journal 2921, where the change of 2902 after time boundary 2903 appears.

[0136] Figure 40A illustrates one read-ahead method. If the system software detects that subsequent read requests from a first read request are sequential relative to contiguous sector runs on disk, the read ahead subsystem may improve performance by extending the read ahead operation to precede the incoming read requests. For example, first request 4000 requests sectors 100 and 101, followed by read requests to sectors 102 and 103 that are fulfilled out of read ahead memory. A second requests 4002 likewise requests sectors 104-107, and a third requests 4004 requests sectors 108-111. The system then detects that sectors are being requested in sequential order, and proceeds without further request to read-ahead sectors 112 through 119, assuming a read ahead window of 8 sectors.

[0137] Figure 40B illustrates another read-ahead method. File systems are able to use much smarter read-ahead since they know which contiguous cluster runs comprise a file. A table 4020 contains the ordered sectors to which a file is written to disk. A first read operation 4022 requests cluster 100, or cluster 0 of the file. The system begins read ahead of cluster 1 of the file, reading cluster 230 on disk. A read request is then received for cluster 230, which is returned from read-ahead memory. The system then performs a read-ahead for the next cluster 2 of the file, or 431 on disk. This operation is carried on so long as new sequential read requests are received and the end of file is not reached. Most file systems, such as NTFS and NWFS perform file level read ahead if they detect sequential access to a file.

[0145] While the present invention has been described and illustrated in conjunction with a number of specific embodiments, those skilled in the art will appreciate that variations and modifications may be made without departing from the principles of the invention as herein illustrated, described and claimed.

Please replace the abstract paragraph with the following paragraph:

[0147] A dual-axis RAID system includes a plurality of X-axis ordinal series of disks, configured to store parity data and a tape drive, and a Y-axis ordinal series of parity disks. The Y-axis series is smaller than the X-axis series, because the X-axis series contains an extra disk configured as a segment journal disk. The RAID system communicates with clients on a network a network via an SCI network interface.